

AD-A129 397

A COMPARISON BETWEEN THE LEAST-SQUARES LATTICE AND FAST 1/1

KALMAN ALGORITHMS..(U) ROYAL SIGNALS AND RADAR

ESTABLISHMENT MALVERN (ENGLAND) J G MCWHIRTER ET AL.

UNCLASSIFIED

MAR 83 RSRE-MEMO-3587 DRIC-BR-87899

F/G 12/1

NL

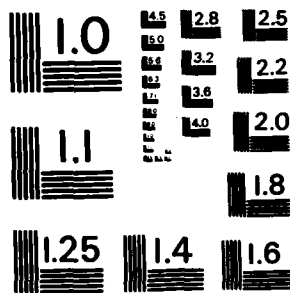
END

DATE

FILED

7 8

E W



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

AD A129397

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3567

TITLE: A COMPARISON BETWEEN THE LEAST-SQUARES LATTICE AND FAST KALMAN ALGORITHMS FOR ADAPTIVE CHANNEL EQUALISATION

AUTHORS: J G McWhirter and T J Shepherd

DATE: March 1983

SUMMARY

↓ An exact least-squares lattice algorithm for channel equalisation is derived without using an explicit time update for the equaliser or predictor coefficients. This avoids the need for auxiliary vectors and scalars within the analysis and facilitates a theoretical comparison with the corresponding Fast Kalman algorithm which is also derived. Both algorithms incorporate a very general form of statistical estimator which includes the growing or growing-fading memory estimators and the sliding window estimator as special cases.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	



Copyright  
C  
Controller HMSO London  
1983

# LIST OF SYMBOLS

$\{a(n)\}$	= primary channel data sequence for noise canceller, or training sequence for equaliser.
$A_p(n)$	= vector of $p^{\text{th}}$ order forward prediction coefficients $a_p(1;n), a_p(2;n), \dots, a_p(p;n)$ at time $n$ .
$\bar{A}_p(n)$	= $(1 \ A_p^T(n))^T$
$B_p(n)$	= vector of $p^{\text{th}}$ backward prediction coefficients $b_p(0;n), b_p(1;n), \dots, b_p(p-1;n)$ at time $n$ .
$\bar{B}_p(n)$	= $(B_p^T(n) \ 1)^T$ .
$C_p(n), D_p(n)$	= $p^{\text{th}}$ order auxiliary vectors at time $n$ .
$e_p(n), e_p(q,n), e_p^i(n), (i=a,b,c,d)$	= forward residuals.
$\bar{e}_p(n), \bar{e}_p(q,n), \bar{e}_p^a(n), \bar{e}_p^b(n)$	= equaliser residuals.
$E_p(n), \hat{E}_p(n)$	= energy of residual $e_p(n)$ .
$\bar{E}_p(n)$	= equaliser residual energy.
$F_p(n)$	= vector of equaliser weights $f_p(0;n), f_p(1;n), \dots, f_p(p;n)$ at time $n$ .
$\bar{F}_p(n)$	= $(1 \ F_p^T(n))^T$ .
$G_p(n)$	= energy of residual $r_p(n)$ .
$k_p(n)$	= $p^{\text{th}}$ order partial autocorrelation function at time $n$ .
$\bar{k}_p(n)$	= $p^{\text{th}}$ order partial cross-correlation function at time $n$ .
$K_p(n), L_p(n)$	= $p^{\text{th}}$ stage forward and backward reflection coefficients at time $n$ .
$L$	= total number of equaliser weights.
$M$	= length of fixed memory.
$M_p(n)$	= component of auxiliary vector $C_{p+1}(n)$ .
$N$	= number of linear prediction stages.
$N_p(n)$	= component of auxiliary vector $D_{p+1}(n)$ .
$p$	= general order of linear prediction or equalisation.
$q_p(n)$	= mean square of $x(n)$ .
$\bar{q}_p(n)$	= mean square of $a(n)$ .

$Q_p(n), V_p(n)$	= autocorrelation vectors of sequence $\{x(n)\}$ .
$\bar{Q}_p(n)$	= cross-correlation vector between $\{x(n)\}$ and $\{a(n)\}$ .
$r_p(n), r_p(q,n), r^i(n), (i=a,b,c,d)$	= backward residuals.
$R_p(n), \hat{R}_p(n)$	= $(p+1) \times (p+1)$ autocorrelation matrix of sequence $\{x(n)\}$ .
$\bar{R}_p(n)$	= extended correlation matrix for $\{x(n)\}$ and $\{a(n)\}$ .
$s_p(n), t_p(n)$	= $(p+1)^{th}$ components of vectors $C_{p+1}(n), D_{p+1}(n)$ , respectively.
$v_p(n), \bar{v}_p(n)$	= mean square of $x(n-p)$ .
$\bar{V}_p(n)$	= extended correlation vector, incorporating cross-correlation between $\{x(n)\}$ and $\{a(n)\}$ .
$w$	= decremental factor for fading memory estimator.
$\{x(n)\}$	= reference channel sequence for noise canceller, or input sequence to equaliser.
$\hat{x}_p(n)$	= $p^{th}$ order forward predicted value for datum $x(n)$ .
$X_p(n)$	= $p^{th}$ order vector for sequence $\{x(n)\}$ .
$\alpha_p(n), \delta_p(n), \gamma_p(n)$	= $p^{th}$ order auxiliary scalars at time $n$ .
$\epsilon_p^e(n), \epsilon_p^r(n)$	= minimised values of energies $E_p(n), G_p(n)$ , respectively.
$\eta$	= conditioning parameter used in initialisation of algorithms.

## 1 INTRODUCTION

The problem of adaptive channel equalisation has received considerable attention in recent signal processing and communications literature and a number of different equalisation algorithms have been proposed. Channel equalisation is the essential mechanism, for example, in noise cancellation systems [1] where the objective is to remove from the signal in the primary channel any components which are correlated with a secondary or reference channel signal. In the case of discrete digital signals this can be achieved by passing the reference signal through an adaptive linear filter whose coefficients are adjusted to minimise the mean-square difference between the filter output and the primary signal. It can be shown that the coefficients, when optimised in this way, satisfy the Wiener-Hopf equation and hence correspond exactly to those of the optimum Wiener filter associated with the process (assumed to be statistically stationary). In some applications the coefficients are only adjusted during an initial 'training period' and are subsequently held constant. In other situations they are updated at regular intervals and the filter may be able to track gradual variations in the channel statistics. In this paper we only consider the case where the coefficients are updated every sample time.

A closely related problem is that of adaptive linear prediction which may be regarded as a special case of channel equalisation in which the reference signal is simply the primary signal delayed by one sample interval. This can also be achieved using a linear filter as described above and the net effect is to decorrelate successive samples of the primary signal leaving, ideally, a white noise sequence. The one-step-ahead linear predictor finds use in a wide range of applications such as speech analysis [2], system identification [3] and maximum entropy spectral analysis [4].

The various algorithms which are commonly used for adaptive channel equalisation may be separated into four main classes according to whether they are based on a transversal or lattice-type filter structure and whether the coefficients are computed exactly or evaluated using gradient or stochastic approximation [3] methods.

In the context of transversal filter structures, stochastic or gradient techniques such as the Widrow "Least Mean Square" (LMS) algorithm [1] are widely used. They are very efficient computationally, requiring a number of arithmetic operations (multiplications or additions per sample time) at most proportional to  $L$ , the number of delay stages in the filter. However, their rate of convergence, which depends on detailed statistical properties of the data, can be slow and the coefficients tend to fluctuate about their converged mean values leading to misadjustment noise and coefficient error. The rate at which the filter can track changes in the statistical environment is therefore limited.

This type of convergence problem is not encountered if the transversal filter coefficients are determined using a direct method such as the "Recursive Least Squares" algorithm, which can be related to the theory of Kalman filtering and has been applied to the channel equalisation problem by Godard [5]. During each sample interval the inverse of the estimated covariance matrix is evaluated by applying an exact update technique. The inverse is then used to derive filter coefficients which are optimal in the sense that they satisfy the Wiener-Hopf equation with respect to the current covariance matrix and cross-correlation vector estimates. By defining the estimators for these quantities to have finite memory length the algorithm can be modified to track temporal variations in the statistical environment. Because of the matrix computations which are involved the number of operations required to carry out the Recursive Least Squares algorithm is proportional to  $L^2$  and, being more expensive



computationally, it is less widely used than algorithms of the gradient type. However, Morf and co-workers [6] and Falconer and Ljung [7] have shown how exact least squares linear prediction and channel equalisation can be performed using a number of arithmetic operations proportional to  $L$  by means of the "Fast Kalman" algorithm, which is very powerful but does not seem to have received much attention in practice.

Much of the attention which has recently been devoted to adaptive channel equalisation concerns the use of lattice-type filter structures which were suggested by Itakura and Saito and can be shown to have very good stability properties based on the theory of orthogonal polynomials [8],[9]. An adaptive lattice filter comprising  $N$  stages can be used to carry out an efficient  $N^{\text{th}}$  order linear prediction for statistically stationary processes by adjusting the reflection coefficient at each stage (using direct or gradient techniques) to ensure that the energy of the residual signal from that stage is minimised. In effect the filter is then being used to implement the Levinson-Durbin (LD) recursion algorithm [10], [11], a generalised version of which is discussed in detail in Section 1.

Griffiths has pointed out that the set of backward residuals derived from an adaptive lattice filter could be used instead of the delayed signal values in a transversal filter when applying gradient techniques to the channel equalisation problem [12]. The fact that these residuals are mutually uncorrelated suggests that the equaliser convergence rate can be greatly improved in general. However, the use of a gradient-type equaliser algorithm will still lead to problems of misadjustment noise and coefficient error and if these are to be kept at an acceptable level it may not be possible in practice to make use of the improved convergence rate which is offered. Furthermore, the behaviour of the lattice filter in the initial convergence phase when the statistics can not be stationary has not been fully analysed in this context.

Morf and Lee [13] have shown that the entire L-stage least-squares channel equalisation process can be carried out exactly using an N-stage lattice-type structure (where  $L = N+1$ ) by generalising the underlying LD recursion algorithm to include the primary channel signal as well as the reference signal and to take account of non-stationary statistics. This approach is extremely powerful since it leads to computationally efficient algorithms (the number of operations is proportional to L) and produces the optimum filter coefficients at every sample time. Moreover, it is based on a filter structure which has good stability properties (at least for stationary statistics) and possesses the desirable property that no stage of the filter affects the operation of any previous stages. Morf and Lee have produced a number of specific algorithms based on this theory and Satorius and Pack [14] have successfully applied one of them to achieve effective equalisation with simulated data on a computer.

In order to develop a complete least-squares lattice algorithm it is necessary to define a suitable estimator for the various statistical quantities which are involved and to include the detailed time evolution of these

estimates within the overall lattice structure. Morf and Lee<sup>[13]</sup> and Satorius and Pack<sup>[14]</sup> have tackled this problem by introducing a number of auxiliary vectors and scalars into the problem and carrying out a lengthy analysis to derive explicit updates in time only for the corresponding transversal filter (equaliser and predictor) coefficients. Satorius and Pack even make use of the Woodbury inversion lemma in their analysis<sup>[15]</sup> and although this move can be avoided for the purpose of deriving their final result it does emphasise the fact that the use of auxiliary vectors and scalars is closely related to the Kalman-Godard approach in which the equaliser coefficients themselves are determined using a direct time update technique<sup>[5]</sup>. It would appear that having solved the Wiener-Hopf equation by means of the efficient time and order recursion which is incorporated in the lattice structure, the solution is effectively being derived again in order to provide time updates for the relevant statistical estimators. This makes the least-squares lattice approach appear much more complicated than it really is although the final algorithm is the same whichever analytic route is chosen.

The purpose of this paper is threefold:

- (a) To demonstrate how the necessary updates can be derived much more readily by means of a very direct analysis which does not involve an explicit time update for the corresponding predictor or equaliser coefficients and leads to more efficient algorithms than those employing auxiliary vectors and scalars;
- (b) To develop a detailed algorithm based on a general estimator for the relevant statistical quantities. This estimator includes as a special case the growing memory (which is used in the Recursive Least Squares algorithm) as well as the growing-fading memory (exponential window)

and the sliding window, both of which provide an important time tracking capability.

- (c) To compare the efficiency and form of these algorithms with their Fast Kalman equivalents.

In Section 2 we present a straightforward derivation of the generalised LD recursion algorithm and show how it leads naturally to the use of lattice structures for linear prediction (Section 2a) and channel equalisation (Section 2b). The analysis is expressed in terms of ensemble averages and serves to provide a clear view of the fundamental way in which the lattice implementation is related to the basic structure of the algorithm. Detailed formulae concerning the estimation and update of the relevant statistical quantities are not necessary for this purpose and so they are discussed separately in Section 3.

In Section 3 a specific estimator for the various statistical quantities is introduced for the first time, and the necessary update formulae to generate complete linear prediction (Section 3a) and channel equalisation (Section 3b) algorithms are derived in a very simple and direct way. The detailed channel equalisation algorithm is given in Appendix 1.

The use of auxiliary vectors and scalars to derive the necessary statistical update formulae is discussed in detail in Section 4a and the detailed equaliser algorithm which results is presented in Appendix 2. The strong resemblance between the auxiliary vector analysis and the use of the Kalman-Godard technique to solve the least-squares channel equalisation problem directly is highlighted in Section 4b where the Fast Kalman algorithm is derived quite simply using some of the auxiliary vector formulae from Section 4a. The Fast Kalman algorithm is given in detail in Appendix 3.

## 2 GENERAL LATTICE AND EQUALISER STRUCTURE

### (a) The $N^{\text{th}}$ Order Linear Prediction Lattice

For the purpose of explaining the underlying structure of lattice equalisation algorithms it is convenient to consider first a general  $p^{\text{th}}$  order linear prediction problem which may be stated as follows. Given a data sequence  $\{x(n) | n = 0, 1, \dots\}$  find the set of time-dependent coefficients or "predictors"  $\{a_p(k;n) | k = 1, 2, \dots, p\}$  which minimises the quantity

$$E_p(n) = \langle e_p^2(n) \rangle \quad (2.1)$$

where  $e_p(n) = x(n) - \hat{x}_p(n)$  is the forward residual and  $\hat{x}_p(n)$  is given by

$$\hat{x}_p(n) = - \sum_{k=1}^p a_p(k;n) x(n-k) \quad (2.2)$$

The angle brackets in equation (2.1) denote a general ensemble average over the non-stationary process  $\{x(n)\}$  which is assumed to be real and scalar. The residual  $e_p(n)$  may be written more concisely as

$$e_p(n) = \bar{A}_p^T(n) X_p(n) \quad (2.3)$$

where we have defined the vectors

$$X_p^T(n) = \left( x(n) \ x(n-1) \ \dots \ x(n-p) \right) \quad (2.4)$$

$$\bar{A}_p^T(n) = \left( 1 \ A_p^T(n) \right) \quad (2.5)$$

$$A_p^T(n) = \left( a_p(1;n) \ a_p(2;n) \ \dots \ a_p(p;n) \right) \quad (2.6)$$

and the superscript T denotes matrix transposition.

If the forward predictors are regarded as non-statistical variables, independent of  $\{x(n)\}$ , equation (2.3) allows the forward residual energy  $E_p(n)$

in (2.1) to be written in the form

$$E_p(n) = \bar{A}_p^T(n) R_p(n) \bar{A}_p(n) , \quad (2.7)$$

where

$$R_p(n) = \langle X_p(n) X_p^T(n) \rangle \quad (2.8)$$

is the  $(p+1) \times (p+1)$  covariance matrix of the process  $\{x(n)\}$ . The  $(i,j)^{th}$  element of the matrix is given by

$$\left[ R_p(n) \right]_{ij} = \langle x(n-i) x(n-j) \rangle \quad (0 \leq i, j \leq p) , \quad (2.9)$$

which is, in general, time dependent. (In the special case where  $\{x(n)\}$  is a stationary process,  $[R_p]_{ij}$  is a function of  $(i-j)$  only and so  $R_p$  is a (symmetric) Toeplitz matrix).

The matrix  $R_p(n)$  possesses useful time-shift properties; from equation (2.9) it is simple to show that the matrix elements satisfy

$$\left[ R_p(n-1) \right]_{ij} = \left[ R_p(n) \right]_{i+1, j+1} ; \quad (2.10)$$

and indeed if the data vector  $X_p(n)$  is partitioned in the form

$$X_p^T(n) = \left( x(n) \mid X_{p-1}^T(n-1) \right) , \quad (2.11)$$

it follows that the covariance matrix has the partitioning

$$R_p(n) = \begin{pmatrix} q_p(n) & Q_p^T(n) \\ Q_p(n) & R_{p-1}(n-1) \end{pmatrix} \quad (2.12)$$

where

$$q_p(n) = \langle x^2(n) \rangle , \quad (2.13)$$

and

$$Q_p(n) = \langle x(n)X_{p-1}(n-1) \rangle . \quad (2.14)$$

An alternative partitioning of  $R_p(n)$  results from writing  $X_p(n)$  in the form

$$X_p^T(n) = \left( X_{p-1}^T(n) \quad x(n-p) \right) . \quad (2.15)$$

from which it follows that

$$R_p(n) = \begin{pmatrix} R_{p-1}(n) & v_p(n) \\ v_p^T(n) & v_p(n) \end{pmatrix} , \quad (2.16)$$

where

$$v_p(n) = \langle x^2(n-p) \rangle ,$$

and

$$v_p(n) = \langle x(n-p)X_{p-1}(n) \rangle . \quad (2.17)$$

Equations (2.5), (2.7) and (2.12) allow the energy  $E_p(n)$  to be written as

$$E_p(n) = q_p(n) + 2A_p^T(n)Q_p(n) + A_p^T(n)R_{p-1}(n-1)A_p(n) , \quad (2.18)$$

and minimisation of  $E_p(n)$  with respect to  $A_p^T(n)$  yields the normal (Yule-Walker) equations for the forward predictors:

$$R_{p-1}(n-1)A_p(n) = -Q_p(n) . \quad (2.19)$$

Substitution of (2.19) into (2.18) gives the minimum energy, or mean-square error,

$$\begin{aligned} \epsilon_p^e(n) &\triangleq \min (E_p(n)) \\ &= q_p(n) + A_p^T(n)Q_p(n) \end{aligned} \quad (2.20)$$

and (2.19) and (2.20) can be combined in the form

$$R_p(n) \bar{A}_p(n) = \begin{pmatrix} \epsilon_p^e(n) \\ 0_p \end{pmatrix}, \quad (2.21)$$

where  $0_p$  is the zero  $p$ -vector.

If equation (2.19) were solved by updating the covariance matrix every sample-time and inverting it directly using a general algorithm such as the Gaussian elimination procedure a large number of arithmetic operations ( $O(p^3)$ ) would be required. Alternatively the inverse covariance matrix itself could be updated every sample time (the Kalman-Godard algorithm [5]) in which case the number of operations would be reduced to  $O(p^2)$ . It is well known, however, that in situations where the statistics are stationary and the covariance matrix is therefore of Toeplitz form the number of operations required can be reduced to  $O(p)$  by updating the solution every sample time using a recursive procedure due to Levinson and Durbin (LD) [10][11]. A point which does not seem to be so widely appreciated is that, by making use of the time-shift properties of  $R_p(n)$  it is possible to generalise this procedure so that it applies to the analysis of time series for which the statistics are not stationary. The generalised LD algorithm will now be derived.

It is necessary to consider in addition the  $p^{\text{th}}$  order backward prediction problem, associated with the minimisation of the quantity

$$G_p(n) = \langle r_p^2(n) \rangle, \quad (2.22)$$

where the backward residual  $r_p(n)$  is defined as

$$r_p(n) = \bar{B}_p^T(n) X_p(n), \quad (2.23)$$

$$\bar{B}_p^T(n) = (B_p^T(n), 1), \quad (2.24)$$

$$\text{and } B_p^T(n) = \begin{pmatrix} b_p(0;n) & b_p(1;n) & \dots & b_p(p-1;n) \end{pmatrix} \quad (2.25)$$

is the vector of backward predictors.



Minimisation of the backward residual energy  $G_p(n)$  with respect to the backward predictors at time  $n$  yields the normal equations

$$R_{p-1}(n)B_p(n) = -v_p(n) \quad (2.26)$$

where the partitioning in (2.16) has been used. In a fashion similar to that used in forward prediction, the normal equations may be expressed in the form

$$R_p(n)\bar{B}_p(n) = \begin{pmatrix} 0 \\ \epsilon_p^r(n) \end{pmatrix}, \quad (2.27)$$

where  $\epsilon_p^r(n)$  is the minimum backward residual energy.

The generalised LD procedure assumes that the  $p^{\text{th}}$ -order forward prediction problem has been solved for time  $n$ , and that the corresponding backward prediction problem has been solved at time  $(n-1)$ ; that is to say that  $A_p(n)$ ,  $B_p(n-1)$ ,  $\epsilon_p^e(n)$ ,  $\epsilon_p^r(n-1)$  are known and satisfy equations (2.21) and (2.27) at the appropriate times. The solution of the  $(p+1)^{\text{th}}$ -order forward and backward prediction problems at time  $n$  may then be written in the form

$$\bar{A}_{p+1}(n) = \begin{pmatrix} \bar{A}_p(n) \\ 0 \end{pmatrix} + K_{p+1}(n) \begin{pmatrix} 0 \\ \bar{B}_p(n-1) \end{pmatrix}, \quad (2.28a)$$

and

$$\bar{B}_{p+1}(n) = L_{p+1}(n) \begin{pmatrix} \bar{A}_p(n) \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \bar{B}_p(n-1) \end{pmatrix}, \quad (2.28b)$$

where  $K_{p+1}(n)$  and  $L_{p+1}(n)$  are the  $(p+1)^{\text{th}}$ -order forward and backward "reflection", or "PARCOR" coefficients. This recursive procedure may be verified by pre-multiplying equations (2.28a) and (2.28b) by  $R_{p+1}(n)$ , partitioned according to equation (2.16) when multiplying  $(\bar{A}_p^T(n) \ 0)^T$ , and according to equation (2.12) when multiplying  $(0 \ \bar{B}_p^T(n-1))^T$ . This yields the relations

$$\epsilon_{p+1}^e(n) = \epsilon_p^e(n) + K_{p+1}(n)\hat{k}_p(n) , \quad (2.29a)$$

$$L_{p+1}(n) = -\frac{\hat{k}_p(n)}{\epsilon_p^e(n)} , \quad (2.29b)$$

$$\epsilon_{p+1}^r(n) = \epsilon_p^r(n) + L_{p+1}(n)k_p(n) , \quad (2.30a)$$

$$K_{p+1}(n) = -\frac{k_p(n)}{\epsilon_p^r(n-1)} , \quad (2.30b)$$

where

$$k_p(n) \triangleq V_{p+1}^T(n)\bar{A}_p(n) , \quad (2.31a)$$

and

$$\hat{k}_p(n) \triangleq Q_{p+1}^T(n)\bar{B}_p(n-1) . \quad (2.31b)$$

The identity

$$k_p(n) = \hat{k}_p(n) = \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} R_{p+1}(n) \begin{pmatrix} \bar{A}_p(n) \\ 0 \end{pmatrix} \quad (2.32)$$

may also be verified by using the two alternative partitionings of  $R_{p+1}(n)$  and taking into account the normal equations (2.21) and (2.27). This equality is often referred to as the "Burg Lemma". Equation (2.32) for  $k_p(n)$  or  $\hat{k}_p(n)$ , together with the definition of  $R_{p+1}(n)$  from equation (2.8), yields

$$\begin{aligned} k_p(n) &= \left\langle \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} X_{p+1}(n) X_{p+1}^T(n) \begin{pmatrix} \bar{A}_p^T(n) & 0 \end{pmatrix}^T \right\rangle \\ &= \langle r_p(n-1) e_p(n) \rangle , \end{aligned} \quad (2.33)$$

where the definitions (2.3) and (2.23) of the forward and backward residuals have been used.  $k_p(n)$  thus has the form of a partial correlation function between the forward and backward residuals, and  $L_p(n)$ ,  $K_p(n)$  (equations (2.29b) and (2.30b)) may be interpreted as partial correlation coefficients.

Equations (2.29), (2.30) and (2.32) provide order recursions for the forward and backward residual energies:

$$\epsilon_{p+1}^e(n) = \epsilon_p^e(n) - \frac{k_p^2(n)}{\epsilon_p^r(n-1)}, \quad (2.34a)$$

$$\epsilon_{p+1}^r(n) = \epsilon_p^r(n-1) - \frac{k_p^2(n)}{\epsilon_p^e(n)}. \quad (2.34b)$$

Recursions for the residuals themselves are found by pre-multiplying equations (2.28) by  $X_{p+1}^T(n)$  and using (2.3) and (2.23), which yields

$$e_{p+1}(n) = e_p(n) + K_{p+1}(n)r_p(n-1), \quad \left( e_0(n) = x(n) \right) \quad (2.35a)$$

and

$$r_{p+1}(n) = L_{p+1}(n)e_p(n) + r_p(n-1), \quad \left( r_0(n) = x(n) \right). \quad (2.35b)$$

The order iterations of equations (2.35) may be realised in the "lattice" or "ladder" network shown in figure 1; they are used to carry out  $N^{\text{th}}$ -order linear prediction by solving the problem recursively for all orders  $p$  ( $0 \leq p \leq N$ ). It can be seen that the solution of the linear prediction problem has been effected using the "local" lattice correlations  $k_p(n)$  rather than the data covariance values, and the reflection coefficients  $L_p(n)$ ,  $K_p(n)$  instead of the usual transversal predictor coefficients  $A_p(n)$ ,  $B_p(n)$ . A knowledge of the  $k_p(n)$  permits a complete evaluation of residuals, reflection coefficients and predictor values using equations (2.35), (2.34), (2.29b), (2.30b) and (2.28).

We note the following points arising from the preceding analysis.

- (i) For stationary statistics the procedure reduces to the conventional LD recursion associated with Toeplitz covariance matrices. In this instance, all quantities become time- ( $n$ )-independent, backward energies equal

forward energies, and backward reflection coefficients equal forward reflection coefficients. Equality for the predictors is also attained in the sense

$$a_p(k) = b_p(p-k) \quad , \quad k = 1, 2, \dots, p. \quad (2.36)$$

It is clear that the more general extended LD recursion is well-suited for situations where it is difficult to realise stationarity - for example, in speech analysis, where local statistics may vary significantly with time, or identification of linear systems with time-dependent parameters. The problem of tracking time-varying characteristics, addressed in Section 3, is one for which the general treatment above is appropriate.

- (ii) The backward residuals are orthogonal. This is easy to demonstrate by evaluating the quantity

$$h_{mp}(n) \triangleq \langle r_m(n) r_p(n) \rangle \quad . \quad (2.37)$$

Using equation (2.23), this may be written

$$h_{mp}(n) = \bar{B}_m^T(n) \langle X_m(n) X_p^T(n) \rangle \bar{B}_p(n) \quad . \quad (2.38)$$

Equations (2.8) and (2.27) show that  $h_{mp}(n)$  is equal to  $\epsilon_p^r(n)$  for  $m = p$ , and zero for  $p < m$ . For  $p > m$  equation (2.27) may be employed in transpose form to prove  $h_{mp}(n)$  zero here also. Hence, we have the orthogonality relation

$$\langle r_m(n) r_p(n) \rangle = \delta_{mp} \epsilon_p^r(n) \quad , \quad (2.39)$$

where

$$\delta_{mp} = \begin{cases} 1, & m = p \\ 0, & m \neq p \end{cases} \quad .$$

It is this property of the lattice filter which Griffiths exploits in order to accelerate the convergence of noise-cancelling filters which operate using gradient techniques such as the LMS algorithm [12].

(b) The  $L^{\text{th}}$  Order Lattice Equaliser

The general  $p^{\text{th}}$  order channel equalisation problem amounts to finding the set of time-dependent equaliser coefficients  $\{f_p(k;n) | k = 0, 1, \dots, p\}$  which minimise the quantity

$$\bar{E}_p(n) \triangleq \langle \bar{e}_p^2(n) \rangle, \quad (2.40)$$

where the residual signal

$$\bar{e}_p(n) = a(n) + \sum_{k=0}^p f_p(k;n)x(n-k)$$

is the difference between the primary signal  $\{a(n)\}$  and the equaliser output obtained by passing the reference signal  $\{x(n)\}$  through the channel equalisation filter.  $\bar{e}_p(n)$  is, in fact, the desired output from a noise cancellation system. This problem may be solved in a recursive manner analogous to that outlined above for linear prediction.

In order to rationalise the notation we define the vector

$$\bar{F}_p^T(n) = \begin{pmatrix} 1 & F_p^T(n) \end{pmatrix}, \quad (2.42)$$

where

$$\bar{F}_p^T(n) = \begin{pmatrix} f_p(0;n) & f_p(1;n) & \dots & f_p(p;n) \end{pmatrix}, \quad (2.43)$$

and the extended data vector

$$\bar{X}_p^T(n) = \begin{pmatrix} a(n) & x_p^T(n) \end{pmatrix}. \quad (2.44)$$

The residual signal  $\bar{e}_p(n)$  may then be written in the form

$$\bar{e}_p(n) = \bar{F}_p^T(n) \bar{X}_p(n) , \quad (2.45)$$

and the error criterion in (2.40) becomes

$$\bar{E}_p(n) = \bar{F}_p^T(n) \bar{R}_p(n) \bar{F}_p(n) , \quad (2.46)$$

where an extended covariance matrix  $\bar{R}_p(n)$  has been defined as

$$\bar{R}_p(n) = \langle \bar{X}_p(n) \bar{X}_p^T(n) \rangle . \quad (2.47)$$

It is easily verified that  $\bar{R}_p(n)$  has the partitioning

$$\bar{R}_p(n) = \begin{pmatrix} \bar{R}_{p-1}(n) & \bar{v}_p(n) \\ \bar{v}_p^T(n) & \bar{v}_p(n) \end{pmatrix} \quad (2.48)$$

where

$$\bar{v}_p(n) = \langle x^2(n-p) \rangle , \quad (2.49)$$

and

$$\bar{v}_p(n) = \langle x(n-p) \bar{X}_{p-1}(n) \rangle . \quad (2.50)$$

It may also be partitioned in the form

$$\bar{R}_p(n) = \begin{pmatrix} \bar{q}_p(n) & \bar{Q}_p^T(n) \\ \bar{Q}_p(n) & R_p(n) \end{pmatrix} , \quad (2.51)$$

where  $\bar{q}_p(n)$  is the mean-square scalar given by

$$\bar{q}_p(n) = \langle a^2(n) \rangle , \quad (2.52)$$

$\bar{Q}_p(n)$  is a cross-correlation vector of the form

$$\bar{Q}_p(n) = \langle a(n) X_p(n) \rangle , \quad (2.53)$$

and  $R_p(n)$  is the covariance matrix defined in equation (2.8).

Use of equations (2.51) and (2.46) permits the minimisation of  $\bar{E}_p(n)$  with respect to  $F_p(n)$ , giving the time-dependent Wiener-Hopf equation,

$$R_p(n)F_p(n) = -\bar{Q}_p(n) \quad (2.54)$$

or

$$\bar{R}_p(n)\bar{F}_p(n) = \begin{pmatrix} \bar{\epsilon}_p(n) \\ 0_{p+1} \end{pmatrix} \quad (2.55)$$

where

$$\bar{\epsilon}_p(n) = \min(\bar{E}_p(n)) \quad (2.56)$$

The extension of the linear prediction lattice filter to a lattice equaliser may be deduced from the equation

$$\bar{F}_p(n) = \begin{pmatrix} \bar{F}_{p-1}(n) \\ 0 \end{pmatrix} - \frac{\bar{k}_p(n)}{\bar{\epsilon}_p^T(n)} \begin{pmatrix} 0 \\ \bar{B}_p(n) \end{pmatrix} \quad (2.57)$$

which relates the equaliser weights to the backward predictor coefficients and may be applied recursively. Equation (2.57) may be verified by pre-multiplying both sides by  $\bar{R}_p(n)$  and using (2.55) and (2.48).  $\bar{k}_p(n)$  is a partial cross-correlation function given by the expression

$$\bar{k}_p(n) = \bar{V}_p^T(n)\bar{F}_{p-1}(n) \quad (2.58)$$

It may also be written in the form

$$\bar{k}_p(n) = \begin{pmatrix} 0 & \bar{B}_p^T(n) \end{pmatrix} \bar{R}_p(n) \begin{pmatrix} \bar{F}_{p-1}^T(n) & 0 \end{pmatrix}^T \quad (2.59)$$

by making use of equation (2.48) and (2.58) and it then follows from (2.47)

that

$$\begin{aligned} \bar{k}_p(n) &= \left\langle \begin{pmatrix} 0 & \bar{B}_p^T(n) \end{pmatrix} \bar{X}_p(n) \bar{X}_p^T(n) \begin{pmatrix} \bar{F}_{p-1}^T(n) & 0 \end{pmatrix}^T \right\rangle \\ &= \langle \bar{r}_p(n) \bar{e}_{p-1}(n) \rangle \end{aligned} \quad (2.60)$$

which takes the form of a partial cross-correlation function between backward and equaliser residuals.

Pre-multiplication of equation (2.57) by  $\bar{X}_p^T(n)$  gives the order recursion for the equaliser residuals

$$\bar{e}_p(n) = \bar{e}_{p-1}(n) - \frac{\bar{k}_p(n)}{\epsilon_p^r(n)} r_p(n) \quad \left( \bar{e}_{-1}(n) = a(n) \right) \quad (2.61)$$

and it follows immediately that the  $N^{\text{th}}$ -order linear prediction lattice filter may be extended to apply to the  $L$  stage channel equalisation/noise-cancellation problem (where  $L = N+1$ ) by means of the configuration illustrated in figure 2 (the equaliser output  $\bar{e}_N(n) - a(n)$  is not specifically shown).

A point worth noting at this stage is that the lattice-type algorithm solves the  $L^{\text{th}}$  order equalisation problem by progressively solving the equivalent problem for all orders  $p \leq L$  and making use of recursions in time and order. Adding extra stages to the filter increases the order of problem which may be solved without affecting the operation of the previous stages in any way. This is to be contrasted with the iterative least-squares methods (eg the Kalman-Godard algorithm) which solve the problem for order  $L$  only by updating the complete coefficient vector solution in time only.

### 3 REALISATION OF AVERAGES

#### (a) Estimator and Linear Prediction

The lattice equaliser recursions of Section 2 constitute a complete algorithm for data processing once suitable estimators for the partial correlation functions  $k_p(n)$  and  $\bar{k}_p(n)$  have been defined. However, as the  $k_p(n)$  and  $\bar{k}_p(n)$  (equations (2.33) and (2.60)) take the form of correlations between various residuals - that is,



filtered data - any explicit estimator for these quantities will necessarily render  $A_p(n)$ ,  $B_p(n)$  and  $F_p(n)$  functions of the data sequence and hence statistical variables, violating our original assumption of deterministic coefficients and predictors. A decoupling of the coefficients and predictors from the data is therefore essential and so an explicit average for the residual energies will be defined as follows. Corresponding estimators for the partial correlation functions will emerge as a result.

The general estimator for the forward energy is chosen here to be

$$\hat{E}_p(n) = \sum_{q=n-M+1}^n w^{n-q} e_p^2(q,n) \quad , \quad (0 < w < 1) \quad (3.1)$$

where the forward residual has been re-defined to be a function of two time epochs  $n$  and  $q$ ,

$$e_p(q,n) = \bar{A}_p^T(n) X_p(q) \quad (x(n) = 0, n < 0) \quad . \quad (3.2)$$

This separation of data and filter evolution permits time averaging over data while the filter parameters are held fixed. Equations (3.1) and (3.2) then yield the expression

$$\hat{E}_p(n) = \bar{A}_p^T(n) \hat{R}_p(n) \bar{A}_p(n) \quad , \quad (3.3)$$

where the estimator for the covariance matrix takes the form

$$\hat{R}_p(n) = \sum_{q=n-M+1}^n w^{n-q} X_p(q) X_p^T(q) \quad , \quad (3.4)$$

and possesses the simple time update recursion,

$$\hat{R}_p(n) = w \hat{R}_p(n-1) + X_p(n) X_p^T(n) - w^M X_p(n-M) X_p^T(n-M) \quad . \quad (3.5)$$

The general estimator described in equations (3.1) to (3.4) includes a number of important special cases such as the growing or growing-fading memory estimator and the sliding window estimator whose specific form and relative merits will be discussed in Section 5. It is likely that the final algorithm would be applied using one of these more specific estimators, although the analysis which follows will be carried out using the general form.

It is easily verified that the time-shift properties (2.12) and (2.16) of the covariance matrix  $R_p(n)$  also apply to  $\hat{R}_p(n)$ ; together with the correspondence between equations (3.3) and (2.7) for the forward energy, this allows the complete lattice structure described in Section 2 to be deployed with the specific estimators described above. For this reason, we shall henceforth omit the caret over estimated quantities. The only alterations to the lattice formalism appear in the definitions and order recursions of residuals. By analogy with equation (3.2), the backward residual is now defined by

$$r_p(q,n) = \bar{B}_p^T(n) X_p(q) \quad , \quad (3.6)$$

and the residual recursions are obtained upon pre-multiplying equations (2.28) by  $X_p^T(q)$ :

$$e_{p+1}(q,n) = e_p(q,n) + K_{p+1}(n) r_p(q-1, n-1) \quad (3.7a)$$

$$r_{p+1}(q,n) = L_{p+1}(n) e_p(q,n) + r_p(q-1, n-1) \quad . \quad (3.7b)$$

It will become clear that only a limited number of these residuals will need to be retained for the final algorithm.

The recursion for the partial correlation function  $k_p(n)$  is now easily derived. We make use of the equation

$$k_p(n) = \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} R_{p+1}(n) \begin{pmatrix} \bar{A}_p^T(n) & 0 \end{pmatrix}^T, \quad (2.32)$$

and difference the forward predictors, giving

$$k_p(n) = \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} R_{p+1}(n) \begin{pmatrix} \bar{A}_p^T(n-1) & 0 \end{pmatrix}^T \\ + \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} R_{p+1}(n) \begin{pmatrix} 0 & \left[ A_p^T(n) - A_p^T(n-1) \right] & 0 \end{pmatrix}^T \quad (3.8)$$

The second term in (3.8) vanishes upon use of equations (2.12) and (2.27), irrespective of  $(A_p(n) - A_p(n-1))$ , and substituting (3.5) (for order  $p+1$ ) into the first term gives

$$k_p(n) = w \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} R_{p+1}(n-1) \begin{pmatrix} \bar{A}_p^T(n-1) & 0 \end{pmatrix}^T \\ + \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} \left[ X_{p+1}(n) X_{p+1}^T(n) - w^M X_{p+1}(n-M) X_{p+1}^T(n-M) \right] \\ \times \begin{pmatrix} \bar{A}_{p+1}^T(n-1) & 0 \end{pmatrix}^T. \quad (3.9)$$

Now, equations (2.16), (2.21) and (2.31a) show that

$$R_{p+1}(n-1) \begin{pmatrix} \bar{A}_p^T(n-1) & 0 \end{pmatrix}^T = \begin{pmatrix} \epsilon_p^e(n-1) & 0_p^T k_p(n-1) \end{pmatrix}, \quad (3.10)$$

and it follows that

$$k_p(n) = w k_p(n-1) + \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} X_{p+1}(n) X_{p+1}^T(n) \begin{pmatrix} \bar{A}_p^T(n-1) & 0 \end{pmatrix}^T \\ - w^M \begin{pmatrix} 0 & \bar{B}_p^T(n-1) \end{pmatrix} X_{p+1}(n-M) X_{p+1}^T(n-M) \begin{pmatrix} \bar{A}_p^T(n-1) & 0 \end{pmatrix}^T. \quad (3.11)$$

Finally, equations (3.2) and (3.6) provide the relationship

$$k_p(n) = w k_p(n-1) + r_p(n-1, n-1) e_p(n, n-1) \\ - w^M r_p(n-M-1, n-1) e_p(n-M, n-1), \quad (3.12)$$

which completes the lattice algorithm for linear prediction. It shows that four types of forward residual ( $e_p(n, n-1)$ ,  $e_p(n, n)$ ,  $e_p(n-M, n-1)$ ,  $e_p(n-M, n)$ ) and their corresponding backward residuals must be carried through the lattice recursions (3.7). Alternative updates for  $k_p(n)$  may be found by differencing the terms in (2.32) in a different order; for example, differencing  $\bar{B}_p(n-1)$  before  $R_{p+1}(n)$  gives

$$k_p(n) = wk_p(n-1) + r_p(n-1, n-2) e_p(n, n) - w^M r_p(n-M-1, n-2) e_p(n-M, n) \quad (3.13)$$

which can be evaluated using the same set of residuals as are required for (3.12). It is noted that many other forms are possible as an arbitrary change in the time argument of either  $\bar{B}_p(n-1)$  or  $\bar{A}_p(n)$  in (2.32) leaves  $k_p(n)$  unchanged.

(b) Estimator and Channel Equalisation

The update for the cross-correlation  $\bar{k}_p(n)$  is as straightforward to derive as that for  $k_p(n)$ . First, the equaliser error is re-defined as

$$\bar{e}_p(q, n) = \bar{F}_p^T(n) \bar{X}_p(q) \quad (3.14)$$

which, from (2.57), has an order recursion

$$\bar{e}_p(q, n) = \bar{e}_{p-1}(q, n) - \frac{\bar{k}_p(n)}{\epsilon_p^r(n)} r_p(q, n) \quad (3.15)$$

The estimator for  $\bar{R}_p(n)$  takes the form

$$\bar{R}_p(n) = \sum_{q=n-M+1}^n w^{n-q} \bar{X}_p(q) \bar{X}_p^T(q) \quad (3.16)$$

and is updated through

$$\bar{R}_p(n) = w\bar{R}_p(n-1) + \bar{X}_p(n)\bar{X}_p^T(n) - w^M\bar{X}_p(n-M)\bar{X}_p^T(n-M) \quad (3.17)$$

From the definition

$$\bar{k}_p(n) = \begin{pmatrix} 0 & \bar{B}_p^T(n) \end{pmatrix} \bar{R}_p(n) \begin{pmatrix} \bar{F}_{p-1}^T(n) & 0 \end{pmatrix}^T, \quad (2.59)$$

a time update is obtained by differencing  $\bar{F}_{p-1}(n)$ , then  $\bar{R}_p(n)$ , in a manner identical to that for  $k_p(n)$  above. Equations (3.17) and (3.14) then give

$$\begin{aligned} \bar{k}_p(n) &= w\bar{k}_p(n-1) + r_p(n, n) \bar{e}_{p-1}(n, n-1) \\ &\quad - w^M r_p(n-M, n) \bar{e}_{p-1}(n-M, n-1), \end{aligned} \quad (3.18)$$

or, alternatively,

$$\begin{aligned} \bar{k}_p(n) &= w\bar{k}_p(n-1) + r_p(n, n-1) \bar{e}_{p-1}(n, n) \\ &\quad - w^M r_p(n-M, n-1) \bar{e}_{p-1}(n-M, n). \end{aligned} \quad (3.19)$$

(3.18) and (3.19) show that only two types of equaliser residual, or "error", need be carried through the algorithm - either  $(\bar{e}_p(n, n-1), \bar{e}_p(n-M, n-1))$  in (3.18) or  $(\bar{e}_p(n, n), \bar{e}_p(n-M, n))$  in (3.19); the necessary lattice residuals are provided in either case.

The complete lattice equaliser algorithm in the form so far described is set out explicitly in Appendix 1 for an equaliser of  $L = N + 1$  taps. A rough count of the number of operations involved for the three types of memory is given in Table 1. Here divisions are counted as multiplications, and repeated operations (for example, computing  $k_p^2(n)$  a second time) are omitted.

<u>Algorithm</u>	<u>Multiplications</u>	<u>Additions</u>
Growing memory	13L - 9	9L - 6
Growing-fading memory	15L - 9	9L - 6
Sliding window	20L - 14	16L - 10

Table 1 Operation count of algorithm in Appendix 1

#### 4 AUXILIARY VARIABLES AND THE FAST KALMAN ALGORITHM

A variant of the algorithm derived in the last two sections is usually found in the literature<sup>[13],[14]</sup>. It requires fewer residuals to update the partial correlation functions, but involves a number of auxiliary variables. The auxiliary vectors and scalars will be derived in Section 4a, and the resulting least-squares lattice algorithm compared with that of Appendix 1. The relations derived in this exercise then form the foundations for the so-called "Fast Kalman" algorithm, which is characterised by purely time-update recursions, and is described in Section 4b.

##### (a) Auxiliary Vectors and Scalars

It was noted in Section 3 that the general update for  $k_p(n)$  in equations (3.12) and (3.13) involves four types of forward and backward residual. It is seen that pairs of these residuals, eg  $(e_p(n,n-1), e_p(n,n))$  differ only in their second time argument, associated with time evolution of the predictor coefficients. These residuals may therefore be related by means of an explicit time update for the predictor coefficients. Such an update is obtained by first differencing the normal equations (2.21) in the form

$$R_p(n)\bar{A}_p(n) - wR_p(n-1)\bar{A}_p(n-1) = \begin{pmatrix} \epsilon_p^e(n) \\ 0_p \end{pmatrix} - w \begin{pmatrix} \epsilon_p^e(n-1) \\ 0_p \end{pmatrix},$$

ie

$$R_p(n) \begin{pmatrix} 0 \\ A_p(n) - A_p(n-1) \end{pmatrix} + \left[ R_p(n) - wR_p(n-1) \right] \bar{A}_p(n-1) = \begin{pmatrix} \epsilon_p^e(n) - w\epsilon_p^e(n-1) \\ 0_p \end{pmatrix} \quad (4.1)$$

Taking the lower  $p$  components of this equation (with the first  $R_p(n)$  partitioned as in (2.12)), and making use of (3.2) and (3.5), we obtain the relationship

$$R_{p-1}(n-1) \begin{pmatrix} A_p(n) - A_p(n-1) \end{pmatrix} = -e_p(n, n-1)X_{p-1}(n-1) + w^M e_p(n-M, n-1)X_{p-1}(n-M-1) \quad (4.2)$$

Auxiliary vectors  $C_p(n)$  and  $D_p(n)$  are now defined, satisfying

$$R_p(n) \begin{pmatrix} C_p(n) \\ D_p(n) \end{pmatrix} = \begin{pmatrix} X_p(n) \\ X_p(n-M) \end{pmatrix} \quad (4.3)$$

If  $R_{p-1}(n-1)$  is non-singular, then equation (4.2) may be multiplied by  $R_{p-1}^{-1}(n-1)$  to provide a time recursion for the forward predictors in terms of the auxiliary vectors,

$$\bar{A}_p(n) = \bar{A}_p(n-1) - e_p(n, n-1) \begin{pmatrix} 0 \\ C_{p-1}(n-1) \end{pmatrix} + w^M e_p(n-M, n-1) \begin{pmatrix} 0 \\ D_{p-1}(n-1) \end{pmatrix} \quad (4.4)$$

Similarly, differencing the backward normal equations gives a time recursion for the backward predictors,

$$\begin{aligned}\bar{B}_p(n) &= \bar{B}_p(n-1) - r_p(n, n-1) \begin{pmatrix} C_{p-1}(n) \\ 0 \end{pmatrix} \\ &+ w^M r_p(n-M, n-1) \begin{pmatrix} D_{p-1}(n) \\ 0 \end{pmatrix} .\end{aligned}\quad (4.5)$$

If auxiliary scalar variables  $\gamma_p(n)$ ,  $\delta_p(n)$ ,  $\alpha_p(n)$  are now defined by the equations

$$\begin{pmatrix} \gamma_p(n) & \delta_p(n) \\ \delta_p(n) & \alpha_p(n) \end{pmatrix} = \begin{pmatrix} X_p^T(n) \\ X_p^T(n-M) \end{pmatrix} R_p^{-1}(n) \begin{pmatrix} X_p(n) & X_p(n-M) \end{pmatrix}, \quad (4.6a)$$

$$= \begin{pmatrix} X_p^T(n) \\ X_p^T(n-M) \end{pmatrix} \begin{pmatrix} C_p(n) & D_p(n) \end{pmatrix}, \quad (4.6b)$$

then relations between  $(e_p(n, n-1), e_p(n-M, n-1))$  and  $(e_p(n, n), e_p(n-M, n-1))$ , etc result from pre-multiplication of (4.4) and (4.5) by  $X_p^T(n)$ ,  $X_p^T(n-M)$ :

$$\begin{pmatrix} e_p(n, n) \\ e_p(n-M, n) \end{pmatrix} = \begin{pmatrix} 1 - \gamma_{p-1}(n-1) & w^M \delta_{p-1}(n-1) \\ -\delta_{p-1}(n-1) & 1 + w^M \alpha_{p-1}(n-1) \end{pmatrix} \begin{pmatrix} e_p(n, n-1) \\ e_p(n-M, n-1) \end{pmatrix}, \quad (4.7a)$$

and

$$\begin{pmatrix} r_p(n, n) \\ r_p(n-M, n) \end{pmatrix} = \begin{pmatrix} 1 - \gamma_{p-1}(n) & w^M \delta_{p-1}(n) \\ -\delta_{p-1}(n) & 1 + w^M \alpha_{p-1}(n) \end{pmatrix} \begin{pmatrix} r_p(n, n-1) \\ r_p(n-M, n-1) \end{pmatrix}. \quad (4.7b)$$

Equations (4.7) allow two pairs of residuals to be eliminated from the lattice equaliser recursions upon the introduction of three auxiliary scalars.



In order to complete this derivation of the least-squares lattice algorithm it is necessary to make use of the following order recursions for the auxiliary vectors.

$$C_p(n) = \begin{pmatrix} C_{p-1}(n) \\ 0 \end{pmatrix} + \frac{r_p(n,n)}{\epsilon_p^r(n)} \bar{B}_p(n) \quad , \quad (4.8)$$

$$D_p(n) = \begin{pmatrix} D_{p-1}(n) \\ 0 \end{pmatrix} + \frac{r_p(n-M,n)}{\epsilon_p^r(n)} \bar{B}_p(n) \quad . \quad (4.9)$$

These may be readily verified by multiplying both sides of the equations by the covariance matrix  $R_p(n)$  partitioned in the appropriate manner.

Finally, order recursions for the auxiliary scalars are obtained by pre-multiplying equations (4.8) and (4.9) by  $X_p^T(n)$ ,  $X_p^T(n-M)$ . The result is as follows,

$$\begin{pmatrix} \gamma_p(n) & \delta_p(n) \\ \delta_p(n) & \alpha_p(n) \end{pmatrix} = \begin{pmatrix} \gamma_{p-1}(n) & \delta_{p-1}(n) \\ \delta_{p-1}(n) & \alpha_{p-1}(n) \end{pmatrix} \quad (4.10)$$

$$+ \frac{1}{\epsilon_p^r(n)} \begin{pmatrix} r_p^2(n,n) & r_p(n-M,n)r_p(n,n) \\ r_p(n,n)r_p(n-M,n) & r_p^2(n-M,n) \end{pmatrix} .$$

In the present context (lattice filters employing auxiliary variables), equation (4.10) completes the alternative formulation. The corresponding algorithm is given explicitly in Appendix 2, where the residuals  $e_p(n,n)$ ,  $r_p(n,n)$ ,  $e_p(n-M,n)$ , and  $r_p(n-M,n)$  have been eliminated (using equations (4.7)) in favour of the set  $(e_p(n,n-1), r_p(n,n-1), e_p(n-M,n-1), r_p(n-M,n-1))$ . A count of operations involved is given in Table 2.

<u>Algorithm</u>	<u>Multiplications</u>	<u>Additions</u>
Growing memory	15L-11	8L-5
Growing-fading memory	17L-11	8L-5
Sliding window	34L-25	21L-13

Table 2. Operation Count of Algorithm in Appendix 2

We note that

- (1) the reduction in the number of residuals achieved by the introduction of auxiliary variables applies only to the linear prediction portion of the equaliser algorithm;
- (2) the saving in terms of number of variables has been traded for a decrease in algorithm efficiency - particularly in the case of the sliding window.

(b) The Fast Kalman Algorithm

Order recursions for auxiliary vectors are the essential ingredients required to derive the Fast Kalman equaliser algorithm [7], which is also a fast, recursive least-squares method, but does not employ a lattice structure; instead it provides direct time updates for the equaliser coefficients  $\bar{F}_p(n)$  required to implement an adaptive transversal filter. For purposes of comparison with the least-squares lattice approach we present a derivation of the Fast Kalman algorithm here.

Differencing equation (2.56) gives

$$\bar{R}_p(n)\bar{F}_p(n) - w\bar{R}_p(n-1)\bar{F}_p(n-1) = \begin{pmatrix} \bar{\epsilon}_p(n) \\ 0_{p+1} \end{pmatrix} - w \begin{pmatrix} \bar{\epsilon}_p(n-1) \\ 0_{p+1} \end{pmatrix} ,$$

or

$$\bar{R}_p(n) \begin{pmatrix} 0 \\ F_p(n) - F_p(n-1) \end{pmatrix} + \left[ \bar{R}_p(n) - w\bar{R}_p(n-1) \right] \bar{F}_p(n-1) = \begin{pmatrix} \bar{e}_p(n) - w\bar{e}_p(n-1) \\ 0_{p+1} \end{pmatrix}. \quad (4.11)$$

When (3.17) is used for the differenced form of  $\bar{R}_p(n)$ , the lower  $(p+1)$  components (with  $\bar{R}_p(n)$  partitioned as in equation (2.51)) of (4.11) provide the relationship

$$R_p(n) \begin{pmatrix} F_p(n) - F_p(n-1) \end{pmatrix} = -\bar{e}_p(n, n-1)X_p(n) + w^M \bar{e}_p(n-M, n-1)X_p(n-M), \quad (4.12)$$

where the definition (3.14) of the equaliser residual  $\bar{e}_p(q, n)$  has been employed. Again assuming the non-singularity of  $R_p(n)$ , (4.12) provides the time recursion for the transversal filter weights, after multiplication by  $R_p^{-1}(n)$ :

$$F_p(n) = F_p(n-1) - \bar{e}_p(n, n-1)C_p(n) + w^M \bar{e}_p(n-M, n-1)D_p(n). \quad (4.13)$$

Equation (4.13) serves as the basis for the adaptive transversal equaliser. The weights in the vector  $F_p(n)$  can be updated in time, provided that residuals  $\bar{e}_p(n, n-1)$ ,  $\bar{e}_p(n-M, n-1)$  and auxiliary vectors  $C_p(n)$ ,  $D_p(n)$  are calculable from quantities known at time  $(n-1)$ , together with the new data values  $x(n)$ ,  $a(n)$ .

As the previous weight vector  $F_p(n-1)$  is assumed known, the requisite residuals may be computed using the formulae

$$\bar{e}_p(n, n-1) = \bar{F}_p^T(n-1)\bar{X}_p(n) = a(n) + F_p^T(n-1)X_p(n) \quad (4.14a)$$

and

$$\bar{\epsilon}_p(n-M, n-1) = \bar{F}_p^T(n-1) \bar{X}_p(n-M) = a(n-M) + F_p^T(n-1) X_p(n-M) \quad (4.14b)$$

To obtain updates for  $C_p(n)$  and  $D_p(n)$ , use is made of equations (4.8) and (4.9) together with the alternative order recursions

$$C_p(n) = \begin{pmatrix} 0 \\ C_{p-1}(n-1) \end{pmatrix} + \frac{e_p(n, n)}{\epsilon_p^e(n)} \bar{A}_p(n) \quad (4.15)$$

$$D_p(n) = \begin{pmatrix} 0 \\ D_{p-1}(n-1) \end{pmatrix} + \frac{e_p(n-M, n)}{\epsilon_p^e(n)} \bar{A}_p(n) \quad (4.16)$$

which may be verified in a similar manner. These relationships (with  $p$  replaced by  $p+1$ ) can obviously be used to generate direct time updates for the auxiliary vectors, provided that the linear prediction quantities involved ( $\bar{A}$ ,  $\bar{B}$ ,  $e$ ,  $r$ ,  $\epsilon$ ) can also be updated. (For example, equation (4.15) is used to form  $C_{p+1}(n)$  from  $C_p(n-1)$ , and thus equation (4.8) may, in principle, be used to extract  $C_p(n)$  from  $C_{p+1}(n)$ ).

Assuming knowledge of past vectors  $A_{p+1}(n-1)$ ,  $B_{p+1}(n-1)$ , and energies  $\epsilon_{p+1}^e(n-1)$ ,  $\epsilon_{p+1}^r(n-1)$ , the time update (4.4) is used in the form

$$\begin{aligned} A_{p+1}(n) &= A_{p+1}(n-1) - e_{p+1}(n, n-1) C_p(n-1) \\ &\quad + w^M e_{p+1}(n-M, n-1) D_p(n-1) \end{aligned} \quad (4.17)$$

to compute  $A_{p+1}(n)$ . This requires forward residuals  $e_{p+1}(n, n-1)$  and  $e_{p+1}(n-M, n-1)$  to have been found from  $A_{p+1}(n-1)$  and the latest data vector using

$$e_{p+1}(n, n-1) = \bar{A}_{p+1}^T(n-1)X_{p+1}(n) = x(n) + A_{p+1}^T(n-1)X_p(n-1) \quad , \quad (4.18a)$$

$$\begin{aligned} e_{p+1}(n-M, n-1) &= \bar{A}_{p+1}^T(n-1)X_{p+1}(n-M) \\ &= x(n-M) + A_{p+1}^T(n-1)X_p(n-M-1) \quad . \end{aligned} \quad (4.13b)$$

A second pair of forward residuals, used in the construction of  $C_{p+1}(n)$ , may now be found,

$$e_{p+1}(n, n) = \bar{A}_{p+1}^T(n)X_{p+1}(n) = x(n) + A_{p+1}^T(n)X_p(n-1) \quad , \quad (4.19a)$$

$$\begin{aligned} e_{p+1}(n, n-M) &= \bar{A}_{p+1}^T(n)X_{p+1}(n-M) \\ &= x(n-M) + A_{p+1}^T(n)X_p(n-M-1) \quad , \end{aligned} \quad (4.19b)$$

while the forward energy  $\epsilon_{p+1}^e(n)$  has an update

$$\begin{aligned} \epsilon_{p+1}^e(n) &= w \epsilon_{p+1}^e(n-1) + \epsilon_{p+1}(n, n-1)e_{p+1}(n, n) \\ &\quad - w^M \epsilon_{p+1}^e(n-M, n-1)e_{p+1}(n-M, n) \quad . \end{aligned} \quad (4.20)$$

Equation (4.20) for  $\epsilon_p^e(n)$  may be obtained by first substituting the differenced matrix,  $(R_p(n) - R_p(n-1))$  from (3.5) into (4.1) to give

$$\begin{aligned} R_p(n) \begin{pmatrix} 0 \\ A_p(n) - A_p(n-1) \end{pmatrix} + e_p(n, n-1)X_p(n) - w e_p(n-M, n-1)X_p(n-M) = \\ \begin{pmatrix} \epsilon_p^e(n) - w \epsilon_p^e(n-1) \\ 0_p \end{pmatrix} \quad , \end{aligned} \quad (4.21)$$

and projecting out the upper component by pre-multiplication with  $\bar{A}_p^T(n)$ , together with use of the normal equation (2.21).  $C_{p+1}(n)$  and  $D_{p+1}(n)$  are now constructed from equations (4.15) and (4.16).

Introduce the notation

$$C_{p+1}(n) = \begin{pmatrix} M_{p+1}(n) \\ s_{p+1}(n) \end{pmatrix}, \quad (4.22a)$$

$$D_{p+1}(n) = \begin{pmatrix} N_{p+1}(n) \\ t_{p+1}(n) \end{pmatrix}, \quad (4.22b)$$

where  $M_{p+1}(n)$  and  $N_{p+1}(n)$  are  $(p+1)$ -vectors, and  $s_{p+1}(n)$ ,  $t_{p+1}(n)$  are scalars; then equations (4.8) and (4.9) (with  $p \rightarrow p+1$ ) may be written in the form

$$C_p(n) = M_{p+1}(n) - s_{p+1}(n)B_{p+1}(n), \quad (4.23a)$$

$$D_p(n) = N_{p+1}(n) - t_{p+1}(n)B_{p+1}(n), \quad (4.23b)$$

where

$$s_{p+1}(n) = \frac{r_{p+1}(n,n)}{\epsilon_{p+1}^r(n)}, \quad (4.24a)$$

$$t_{p+1}(n) = \frac{r_{p+1}(n-M,n)}{\epsilon_{p+1}^r(n)}. \quad (4.24b)$$

As  $M_{p+1}(n)$ ,  $N_{p+1}(n)$ ,  $s_{p+1}(n)$ ,  $t_{p+1}(n)$  are known components for  $C_{p+1}(n)$ , we require only an update for  $B_{p+1}(n)$  to compute  $C_p(n)$ ,  $D_p(n)$  in equations (4.23). Equation (4.5) may be rewritten as

$$B_{p+1}(n) = B_{p+1}(n-1) - r_{p+1}(n, n-1)C_p(n) + w^M r_{p+1}(n-M, n-1)D_p(n) \quad , \quad (4.25)$$

which is seen to depend in turn on  $C_p(n)$ ,  $D_p(n)$ . However, substitution of  $B_{p+1}(n)$  from (4.23) into (4.25) yields

$$\begin{aligned} B_{p+1}(n) & \left( 1 - r_{p+1}(n, n-1) \frac{r_{p+1}(n, n)}{\epsilon_{p+1}(n)} + w^M r_{p+1}(n-M, n-1) \frac{r_{p+1}(n-M, n)}{\epsilon_{p+1}(n)} \right) \\ & = B_{p+1}(n-1) - r_{p+1}(n, n-1)M_{p+1}(n) + w^M r_{p+1}(n-M, n-1)N_{p+1}(n) \quad , \end{aligned}$$

ie

$$B_{p+1}(n) = \frac{\left( B_{p+1}(n-1) - r_{p+1}(n, n-1)M_{p+1}(n) + w^M r_{p+1}(n-M, n-1)N_{p+1}(n) \right)}{\left( 1 - r_{p+1}(n, n-1)s_{p+1}(n) + w^M r_{p+1}(n-M, n-1)t_{p+1}(n) \right)} \quad , \quad (4.26)$$

which may be evaluated after the backward residuals have been computed from

$$r_{p+1}(n, n-1) = x(n-p-1) + B_{p+1}^T(n-1)X_p(n) \quad , \quad (4.27a)$$

$$r_{p+1}(n-M, n-1) = x(n-M-p-1) + B_{p+1}^T(n-1)X_p(n-M) \quad . \quad (4.27b)$$

The auxiliary vectors  $C_p(n)$ ,  $D_p(n)$  can now be obtained from equations (4.23), and hence the equaliser weights from (4.13). The complete algorithm is given explicitly in Appendix 3.

The following points are to be noted.

- (i) The Fast Kalman algorithm provides linear prediction residuals  $e_{p+1}$  and  $r_{p+1}$  as a by-product of the equalisation computation; in particular, it may be configured as three transversal filters, executing finite convolution of the data with the vectors  $F_p$ ,  $A_{p+1}$ , and  $B_{p+1}$ .

- (ii) The algorithm residuals,  $e$ ,  $r$ , and  $\bar{e}$  are arithmetically equivalent to corresponding residuals of the lattice equaliser, as both algorithms perform exact Recursive Least Squares estimation.
- (iii) The algorithm is not "expandable" in the same sense as that employing a lattice configuration; that is to say that for a transversal filter configuration, increasing the number of weights affects all previous stages; this may give rise to different numerical properties between the Fast Kalman and lattice algorithms.
- (iv) Auxiliary scalars may be introduced into the algorithm, eliminating a number of predictor residuals, in a manner similar to that in Section 4a.
- (v) As in the case of the lattice equaliser, the data vector  $X_{p+M}(n)$ , as well as the sequence  $(a(n), a(n-1), \dots, a(n-M))$ , must be stored for the sliding window estimator, and  $X_p(n)$  only for the growing-memory estimator.
- (vi) A count of operations involved in the algorithm given in Appendix 3 is provided in Table 3. It is marginally more efficient than the many-residual lattice equaliser algorithm.

<u>Algorithm</u>	<u>Multiplications</u>	<u>Additions</u>
Growing memory	$10L + 3$	$9L + 2$
Growing-fading memory	$10L + 4$	$9L + 2$
Sliding window	$19L + 6$	$18L + 4$

Table 3 Operation Count of Fast Kalman Algorithm in  
Appendix 3



## 5 COMMENTS AND CONCLUSIONS

In this paper we have derived two alternative fast algorithms for performing exact least-squares linear prediction and channel equalisation. The least-squares lattice algorithm is implemented using a lattice filter structure whose coefficients are evaluated using time and order updates, whereas the Fast Kalman algorithm is implemented using a transversal filter whose coefficients are determined using updates in time only. Both of the algorithms incorporate a general statistical estimator which can be used to track variations in the local statistics or to converge to an assumed stationary value. It includes the following special cases which merit some discussion.

- i) When  $M$  is infinite and  $w = 1$  all data points from zero to  $n$  are equally weighted. We call this the "growing" memory. (The data pre-windowing condition in (3.2) sets the lower limit of summation at  $q = 0$  in (3.1) and (3.4)).
- ii) When  $M$  is infinite and  $w < 1$  the average has an effective memory length of  $-1/\log_e w$  and registers predominantly local statistics. We call this the "growing-fading" memory.
- iii) In the limit  $w = 1$ , the average possesses a memory of  $M$  time units, and hence registers only local statistics within that interval. We refer to this as the "sliding window" memory.

The growing memory estimator provides exact algorithms arithmetically equivalent to a Recursive Least Squares procedure and is suitable for obtaining a fixed parameter filter in situations where the statistics of the input data are assumed to be stationary. The adaptation rate decreases with data running time, and the filter becomes insensitive to changes in data correlation; it is therefore likely to prove unsuitable for use in processing long data sequences. In addition, the forward energies (as defined above) necessarily increase with

time, although overflow may be avoided fairly simply by forming averages involving division by the time elapsed. The growing-fading memory and sliding window estimator can track local variations in data statistics and may be used when non-stationarity is assumed; this is the regime for which the generalised LD recursion is most appropriate. The growing-fading memory is equivalent to an exponential window, while the sliding window has a fixed memory of  $M$  units, corresponding to a rectangular window within which all data points are assigned an equal weight. The growing-fading memory algorithms are seen to be far more efficient than their sliding window counterparts, which must keep track of the trailing edge of the window.

The actual estimator for the correlation matrix is

$$R_p(n) = \sum_{q=n-M+1}^n w^{n-q} X_p(q) X_p^T(q) + \eta w^{n+1} I_{p+1}, \quad (5.1)$$

where  $I_p$  is a  $p \times p$  unit matrix. It is evident that the effect of the parameter  $n$ , which renders the algorithms well-conditioned, is "forgotten" by the growing estimator after times beyond the memory length, but is retained for all times by the sliding window estimator. Although the conditioning parameter causes a deviation from optimum least-squares adaptation its loss of influence after a given time may be considered a disadvantage, especially in cases where the residual energies drop below the available computational accuracy. In this instance conditioning may have to be reinstated unless the sliding window algorithm is employed. In either estimator accuracy is limited by residual statistical fluctuations associated with finite time averaging, by the effect of the window lagging behind the present data input, and by the frequency discrimination which arises from apodisation.

Since the least-squares lattice and Fast Kalman algorithms compute an exact solution to the same least-squares problem they must be arithmetically

equivalent. However they are quite different in structure and this is fundamentally associated with the fact that the equaliser coefficients in the Fast Kalman algorithm are explicitly updated in time only, whereas the coefficient in the least-squares lattice algorithm are updated using recursions in time and order. The derivation of the least-squares lattice algorithm presented in this paper highlights this distinction by avoiding the use of an explicit time update for the predictor coefficients when generating a time update for the reflection coefficient estimator. This, in turn, avoids the need to introduce auxiliary vectors and scalars into the least-squares lattice analysis. The auxiliary variables are required, however, in the derivation of the Fast Kalman algorithm where an explicit time update for the equaliser coefficients is essential and can be derived efficiently by making use of the time and order recursive properties of the auxiliary vectors, equations (4.8), (4.9), (4.15) and (4.16). It is interesting to note that all of these relationships together with those which are required for the generalised LD recursion can be derived fairly simply by using the following covariance matrix inversion lemma

$$R_p^{-1}(n) = \begin{pmatrix} 0 & 0_p^T \\ 0_p & R_{p-1}^{-1}(n-1) \end{pmatrix} + \frac{\bar{A}_p(n)\bar{A}_p^T(n)}{\epsilon_p^e(n)} \quad (5.2a)$$

$$= \begin{pmatrix} R_{p-1}^{-1}(n) & 0_p \\ 0_p^T & 0 \end{pmatrix} + \frac{\bar{B}_p(n)\bar{B}_p^T(n)}{\epsilon_p^r(n)} \quad (5.2b)$$

Equation (5.2) applies to the specific estimator  $\hat{R}_p(n)$  as well as the ensemble average and can be deduced quite simply by inverting equations (2.12) and (2.16) directly and making use of the normal equations (2.21) and (2.27).

The Fast Kalman algorithm has been shown to be marginally more efficient (in terms of arithmetic operations) than the least-squares lattice algorithm, but is reported to be relatively unstable [16]. The least-squares lattice algorithm on the other hand has a number of desirable numerical properties which arise from the way in which it progressively reduces the residual energy in a stage-by-stage manner. The internal accuracy or wordlength requirements for this type of calculation are generally less stringent, and the stage-by-stage property ensures that the optimum number of stages can be determined by inspection as the calculation progresses. The least-squares lattice algorithm is likely, therefore, to have superior stability properties. However, the stability of adaptive algorithms in situations of non-stationary statistics is much more difficult to analyse and less well understood than those which operate in a stationary environment. For example in the non-stationary case the stability of a lattice filter is not determined by whether or not the reflection coefficients have magnitude less than unity, or the poles of the corresponding transfer function lie inside the unit circle. A detailed investigation of the stability properties of these algorithms would be extremely valuable and is likely to be the subject of much further research.

## APPENDIX 1

### THE LATTICE EQUALISER ALGORITHM

#### 1) Define the scalars

$$i) \quad \epsilon_p^e(n), \quad \epsilon_p^r(n), \quad \epsilon_p^r(n-1); \quad e_p^i(n), \quad r_p^i(n), \quad r_p^i(n-1) \quad (i = a, b, c, d) \quad ;$$

$$\bar{e}_p^a(n), \quad \bar{e}_p^b(n), \quad \bar{k}_p(n), \quad (p = 0, 1, \dots, N)$$

{Comment: Residuals  $\{e_p^i(n)\}$  are to be identified with those of Section 3 in the following manner,  $(e_p^a(n) e_p^b(n) e_p^c(n) e_p^d(n)) = (\bar{e}_p(n, n-1) e_p(n-M, n-1) e_p(n, n) e_p(n-M, n))$ , with a similar correspondence for the  $\{r_p^i(n)\}$ , and  $(\bar{e}_p^a(n) \bar{e}_p^b(n)) = (\bar{e}_p(n, n-1) \bar{e}_p(n-M, n-1))$ .}

$$ii) \quad k_p(n), \quad L_{p+1}(n), \quad K_{p+1}(n) \quad (p = 0, 1, \dots, N-1).$$

#### 2) Initialise the algorithm as follows,

$$i) \quad \epsilon_p^e(-1) = \epsilon_p^r(-1) = \epsilon_p^r(-2) = \eta \quad (p = 0, 1, \dots, N)$$

{Comment: This assignment renders the initial covariance matrix diagonal, with entries equal to  $\eta$ , which may be set equal to a small, positive value.}

$$ii) \quad r_p^i(-1) \quad (i = a, b, c, d) = k_p(-1) = \bar{k}_p(-1) = K_p(-1) = L_p(-1) = 0, \quad (\text{all } p).$$

3) For each successive value of  $n$ , starting at  $n = 0$ , and before iteration over  $p$ , assign the following values to  $p = 0$  parameters.

$$e_0^a(n) = e_0^c(n) = r_0^a(n) = r_0^c(n) = x(n)$$

$$e_0^b(n) = e_0^d(n) = r_0^b(n) = r_0^d(n) = x(n-M)$$

with  $x(n) = 0$  for  $n < 0$

$$\bar{e}_{-1}^a(n) = a(n) \quad , \quad \bar{e}_{-1}^b(n) = a(n-M)$$

$$\epsilon_0^e(n) = \epsilon_0^r(n) = w \epsilon_0^e(n-1) + x^2(n) - w^M x^2(n-M) \quad .$$

{Comment: This last recursion derives from equation (4.20) for  $p+1 = 0$ }.

4) For each value of  $n$ , perform the following iterations for  $p$  from 0 to  $N - 1$ .

$$\begin{pmatrix} e_{p+1}^a(n) \\ e_{p+1}^b(n) \end{pmatrix} = \begin{pmatrix} e_p^a(n) \\ e_p^b(n) \end{pmatrix} + K_{p+1}(n-1) \begin{pmatrix} r_p^a(n-1) \\ r_p^b(n-1) \end{pmatrix} \quad ,$$

$$\begin{pmatrix} r_{p+1}^e(n) \\ r_{p+1}^b(n) \end{pmatrix} = L_{p+1}(n-1) \begin{pmatrix} e_p^a(n) \\ e_p^b(n) \end{pmatrix} + \begin{pmatrix} r_p^a(n-1) \\ r_p^b(n-1) \end{pmatrix} \quad ,$$

$$k_p(n) = w k_p(n-1) + r_p^c(n-1) e_p^a(n) - w^M r_p^d(n-1) e_p^b(n)$$

$$\epsilon_{p+1}^e(n) = \epsilon_p^e(n) - \frac{k_p^2(n)}{\epsilon_p^r(n-1)}$$

$$\epsilon_{p+1}^r(n) = \epsilon_p^r(n-1) - \frac{k_p^2(n)}{\epsilon_p^e(n)}$$

$$K_{p+1}(n) = - \frac{k_p(n)}{\epsilon_p^r(n-1)}$$

$$L_{p+1}(n) = - \frac{k_p(n)}{\epsilon_p^e(n)}$$

$$\begin{pmatrix} e_{p+1}^c(n) \\ e_{p+1}^d(n) \end{pmatrix} = \begin{pmatrix} e_p^c(n) \\ e_p^d(n) \end{pmatrix} + K_{p+1}(n) \begin{pmatrix} r_p^c(n-1) \\ r_p^d(n-1) \end{pmatrix}$$

$$\begin{pmatrix} r_{p+1}^c(n) \\ r_{p+1}^d(n) \end{pmatrix} = L_{p+1}(n) \begin{pmatrix} e_p^c(n) \\ e_p^d(n) \end{pmatrix} + \begin{pmatrix} r_p^c(n-1) \\ r_p^d(n-1) \end{pmatrix}$$

5) Perform, for  $p$  from 0 to  $N$ , the equaliser recursions

$$\begin{pmatrix} \bar{e}_p^a(n) \\ \bar{e}_p^b(n) \end{pmatrix} = \begin{pmatrix} \bar{e}_{p-1}^a(n) \\ \bar{e}_{p-1}^b(n) \end{pmatrix} - \frac{\bar{k}_p(n-1)}{\epsilon_p^r(n-1)} \begin{pmatrix} r_p^a(n) \\ r_p^b(n) \end{pmatrix},$$

$$\bar{k}_p(n) = w \bar{k}_p(n-1) + r_p^c(n) \bar{e}_{p-1}^a(n) - w^M r_p^d(n) \bar{e}_{p-1}^b(n).$$

The output of the predictor may be taken to be either  $e_N^a(n)$  or  $e_N^c(n)$ , and that of the equaliser as  $\bar{e}_N^a(n)$ .

The growing-fading memory algorithm is obtained by setting to zero the residuals  $e^b, e^d, r^b, r^d$  in 1) to 5) above. (This effectively omits any explicit functions of  $M$ , representing the lower limit of the window.) The growing memory algorithm may then be obtained by setting  $w = 1$ .

For the sliding window algorithm,  $w$  is set equal to unity in 1) to 5), and all residuals retained. (In none of these cases does multiplication by  $w^M$  need to be performed.) This last algorithm is arithmetically equivalent to [17] that of Ahmed et al, who employed a Woodbury inversion lemma to update the covariance matrix.

## APPENDIX 2

### THE AUXILIARY VARIABLE LATTICE EQUALISER ALGORITHM

1) Define the scalars

$$\text{i) } \epsilon_p^e(n), \epsilon_p^r(n), \epsilon_p^r(n-1); e_p^a(n), e_p^b(n), r_p^a(n), r_p^b(n), r_p^a(n-1), r_p^b(n-1), \\ \bar{e}_p^a(n), \bar{e}_p^b(n), \bar{k}_p(n), \quad (p = 0, 1, \dots, N)$$

$$\text{ii) } k_p(n), L_{p+1}(n), K_{p+1}(n), \gamma_p(n), \delta_p(n), \alpha_p(n) \quad (p = 0, 1, \dots, N-1)$$

2) Initialise as follows

$$\text{i) } \epsilon_p^e(-1) = \epsilon_p^r(-1) = \epsilon_p^r(-2) = \eta \quad (p = 0, 1, \dots, N)$$

$$\text{ii) } r_p^e(-1) = r_p^b(-1) = k_p(-1) = \bar{k}_p(-1) = K_p(-1) = L_p(-1) = \gamma_p(-1) = \delta_p(-1) \\ = \alpha_p(-1) = 0 \quad (\text{all } p)$$

iii) Retain the lowest-order auxiliary variables for all  $n$ ,

$$\gamma_{-1}(n) = \delta_{-1}(n) = \alpha_{-1}(n) = 0$$

3) For each successive value of  $n$ , starting at  $n = 0$ , and before iteration over  $p$ , assign the following values to  $p = 0$  parameters

$$\left. \begin{aligned} e_0^a(n) &= r_0^a(n) = x(n) \\ e_0^b(n) &= r_0^b(n) = x(n-M) \end{aligned} \right\} \text{ with } x(n) = 0 \text{ for } n < 0$$

$$\bar{e}_{-1}^a(n) = a(n), \quad \bar{e}_{-1}^b(n) = a(n-M)$$

$$\epsilon_0^e(n) = \epsilon_0^r(n) = w \epsilon_0^e(n-1) + x^2(n) - w^M x^2(n-M)$$



4) For each value of  $n$ , perform the following iterations from  $p$  from 0 to  $N-1$ .

$$\begin{pmatrix} e_{p+1}^a(n) \\ e_{p+1}^b(n) \end{pmatrix} = \begin{pmatrix} e_p^a(n) \\ e_p^b(n) \end{pmatrix} + K_{p+1}(n-1) \begin{pmatrix} r_p^a(n-1) \\ r_p^b(n-1) \end{pmatrix},$$

$$\begin{pmatrix} r_{p+1}^a(n) \\ r_{p+1}^b(n) \end{pmatrix} = L_{p+1}(n-1) \begin{pmatrix} e_p^a(n) \\ e_p^b(n) \end{pmatrix} + \begin{pmatrix} r_p^a(n-1) \\ r_p^b(n-1) \end{pmatrix},$$

$$k_p(n) = wk_p(n-1) + \begin{pmatrix} e_p^a(n) & e_p^b(n) \end{pmatrix} \begin{pmatrix} 1-\gamma_{p-1}(n-1) & w_{\delta_{p-1}}^M(n-1) \\ w_{\delta_{p-1}}^M(n-1) & -w^M(1+w_{\alpha_{p-1}}^M(n-1)) \end{pmatrix}$$

$$\times \begin{pmatrix} r_p^a(n-1) \\ r_p^b(n-1) \end{pmatrix},$$

$$\epsilon_{p+1}^e(n) = \epsilon_p^e(n) - \frac{k_p^2(n)}{\epsilon_p^r(n-1)},$$

$$\epsilon_{p+1}^r(n) = \epsilon_p^r(n-1) - \frac{k_p^2(n)}{\epsilon_p^e(n)},$$

$$K_{p+1}(n) = - \frac{k_p(n)}{\epsilon_p^r(n-1)},$$

$$L_{p+1}(n) = - \frac{k_p(n)}{\epsilon_p^e(n)},$$

$$\gamma_p(n) = \gamma_{p-1}(n) + \frac{1}{\epsilon_p^r(n)} \left[ \left( 1 - \gamma_{p-1}(n) \right) r_p^a(n-1) + w^M \delta_{p-1}(n) r_p^b(n-1) \right]^2 ,$$

$$\begin{aligned} \delta_p(n) &= \delta_{p-1}(n) + \frac{1}{\epsilon_p^r(n)} \left[ \left( 1 - \gamma_{p-1}(n) \right) r_p^a(n-1) + w^M \delta_{p-1}(n) r_p^b(n-1) \right] \\ &\quad \times \left[ -\delta_{p-1}(n) r_p^e(n-1) + \left( 1 + w^M \alpha_{p-1}(n) \right) r_p^b(n-1) \right] , \\ \alpha_p(n) &= \alpha_{p-1}(n) + \frac{1}{\epsilon_p^r(n)} \left[ -\delta_{p-1}(n) r_p^e(n-1) + \left( 1 + w^M \alpha_{p-1}(n) \right) r_p^b(n-1) \right]^2 . \end{aligned}$$

5) Perform, for  $p$  from 0 to  $N$ , the equaliser recursions

$$\begin{pmatrix} \bar{e}_p^a(n) \\ \bar{e}_p^b(n) \end{pmatrix} = \begin{pmatrix} \bar{e}_{p-1}^a(n) \\ \bar{e}_{p-1}^b(n) \end{pmatrix} - \frac{\bar{k}_p(n-1)}{\epsilon_p^r(n-1)} \begin{pmatrix} r_p^a(n) \\ r_p^b(n) \end{pmatrix} ,$$

$$\begin{aligned} \bar{k}_p(n) &= w \bar{k}_p(n-1) + \begin{pmatrix} \bar{e}_{p-1}^a(n) & \bar{e}_{p-1}^b(n) \end{pmatrix} \begin{pmatrix} 1 - \gamma_{p-1}(n) & w^M \delta_{p-1}(n) \\ w^M \delta_{p-1}(n) & -w^M (1 + w^M \alpha_{p-1}(n)) \end{pmatrix} \\ &\quad \times \begin{pmatrix} r_p^a(n) \\ r_p^b(n) \end{pmatrix} . \end{aligned}$$

The predictor output is  $e_N^a(n)$ , and the equaliser output  $\bar{e}_N^a(n)$ .

Omit  $e^b$ ,  $r^b$ ,  $\bar{e}^b$  for growing-fading memory algorithm; in addition, set  $w = 1$  for growing memory algorithm. Set  $w = 1$  and retain all residuals for sliding memory algorithm. (As in Appendix 2,  $w^M$  will never appear.)

It is clearly more efficient to work with auxiliary variables  $(1 - \gamma_p(n))$  and  $(1 + w^M \alpha_p(n))$  instead of  $\gamma_p(n)$  and  $\alpha_p(n)$  wherever they occur in 1) to 5) above.

### APPENDIX 3

#### FAST KALMAN ALGORITHM

1) Define the  $(N + 1)$ -vectors

i)  $F(n)$ ,  $C(n)$ ,  $D(n)$ ,  $A(n)$ ,  $B(n)$ ,  $M(n)$ ,  $N(n)$ ,

and the scalars

ii)  $e^a(n)$ ,  $e^b(n)$ ,  $e^c(n)$ ,  $e^d(n)$ ,  $r^a(n)$ ,  $r^b(n)$ ,  $\bar{e}^a(n)$ ,  $\bar{e}^b(n)$ ,  $\epsilon^e(n)$ ,  $s(n)$ ,  $t(n)$

{Comment: these variables correspond to similar expressions in Section 4, for the single value of  $p = N$ }.

2) Initialise as follows

$$\epsilon^e(-1) = \eta$$

$$A(-1) = C(-1) = D(-1) = B(-1) = F(-1) = 0.$$

3) For each value of  $n$ , starting at 0, perform the following iterations.

$$e^a(n) = x(n) + A^T(n-1)X_N(n-1)$$

$$e^b(n) = x(n-M) + A^T(n-1)X_N(n-M-1)$$

$$A(n) = A(n-1) - e^a(n)C(n-1) + w^M e^b(n)D(n-1)$$

{Comment: This recursion is equation (4.4) of the text.}

$$e^c(n) = x(n) + A^T(n)X_N(n-1)$$

$$e^d(n) = x(n-M) + A^T(n)X_N(n-M-1)$$

$$\epsilon^e(n) = w \epsilon^e(n-1) + e^a(n)e^c(n) - w^M e^b(n)e^d(n)$$

{Comment: This is the forward residual energy update of equation (4.20).}

$$\begin{pmatrix} M(n) \\ s(n) \end{pmatrix} = \begin{pmatrix} 0 \\ C(n-1) \end{pmatrix} + \frac{e^c(n)}{\epsilon^e(n)} \begin{pmatrix} 1 \\ A(n) \end{pmatrix},$$

$$\begin{pmatrix} N(n) \\ t(n) \end{pmatrix} = \begin{pmatrix} 0 \\ D(n-1) \end{pmatrix} + \frac{e^d(n)}{\epsilon^e(n)} \begin{pmatrix} 1 \\ A(n) \end{pmatrix},$$

$$r^a(n) = x(n-N-1) + B^T(n-1)X_N(n),$$

$$r^b(n) = x(n-M-N-1) + B^T(n-1)X_N(n-M),$$

$$B(n) = \left( B(n-1) - r^a(n)M(n) + w^M r^b(n)N(n) \right) \cdot \left( 1 - r^a(n)s(n) + w^M r^b(n)t(n) \right)^{-1}$$

$$C(n) = M(n) - s(n)B(n),$$

$$D(n) = N(n) - t(n)B(n).$$

{Comment: This completes the prediction filter algorithm.}

$$4) \quad \bar{e}^a(n) = a(n) + F^T(n-1)X_N(n),$$

$$\bar{e}^b(n) = a(n-M) + F^T(n-1)X_N(n-M),$$

$$F(n) = F(n-1) - \bar{e}^a(n)C(n) + w^M \bar{e}^b(n)D(n).$$

The prediction filter output is taken to be  $e^a(n)$  or  $e^c(n)$ , and that of the equaliser as  $\bar{e}^a(n)$ .

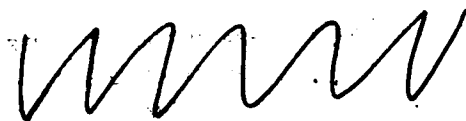
The specialisations for each of the growing, growing-fading, or sliding window algorithms are precisely as described in Appendix 1.

## REFERENCES

- 1 Widrow, B., in "Aspects of Network and System Theory", R Kalman and N DeClaris, Eds, New York: Holt, Rinehart and Winston, 1971, p563-587; Widrow, B., Glover, J R, Jr., McCool, J M., Kaunitz, J., Williams, C S., Hearn, R H., Zeidler, J R., Dong, E, Jr., and Goodlin, R C. "Adaptive Noise Cancelling: Principles and Applications", Proc IEEE, 1975, 63, p 1692-1716.
- 2 Markel, J D and Gray, A H, Jr., "Linear Prediction of Speech", Springer Verlag, Heidelberg, 1976.
- 3 <sup>o</sup> Åström, K J., and Eykhoff, P. "System Identification - A Survey", Automatica 1971, 7, p 123-162.
- 4 Burg, J P. "Maximum Entropy Spectral Analysis", PhD Thesis, Stanford University, Stanford, CA., May 1975.
- 5 Godard, D. "Channel Equalization Using a Kalman Filter for Fast Data Transmission", IBM Journal of Research and Development, May 1974, p 267-273.
- 6 Morf, M., Kailath, T., and Ljung, L. "Fast Algorithms for Recursive Identification", Proc. IEEE Conf. in Decision and Control, Clearwater Beach, Florida, December 1976, p 916-921; Morf, M and Ljung L. "Fast Algorithms for Recursive Identification", IEEE International Symposium on Information Theory, June 1976, Ronneby, Sweden, p 140-141; Ljung, L., Morf, M and Falconer, D D. "Fast Calculation of Gain Matrices for Recursive Estimation Schemes", International J. Control, January 1978, p 1-19.

- 7 Falconer, D D., and Ljung L. "Application of Fast Kalman Estimation to Adaptive Equalization", IEEE Trans., 1978, COM-26, p 1439-1446.
- 8 Itakura, F., and Saito, S. "Digital Filtering Techniques for Speech Analysis and Synthesis", in Proc. 7<sup>th</sup> Int. Cong. Acoust., Budapest, 1971, Paper 25-C-1, p 261-264.
- 9 Markel, J D., and Gray, A H, Jr., "Roundoff Noise Characteristics of a Class of Orthogonal Polynomial Structures", IEEE Trans., 1975, ASSP-23, p 473-486.
- 10 Levinson, N., "The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction", J. Math. Phys., 1947, 25, p 261-278; also Appendix B in Wiener, N., "Extrapolation, Interpolation and Smoothing of Stationary Time Series", Cambridge, Mass. MIT Press, 1949.
- 11 Durbin, J., "The Fitting of Time-Series Models", Rev. Inst. Int. Statist., 1960, 28, p 233-243.
- 12 Griffiths, L J., "An Adaptive Lattice Structure for Noise-Cancelling Applications", in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Tulsa, OK., April, 1978, p 87-90.
- 13 Morf, M and Lee, D T., "Recursive Least Squares Ladder Forms for Fast Parameter Tracking", in Proc. IEEE Conf. on Decision and Control, San Diego, CA, January 1979, p 1362-1367.
- 14 Satorius, E H., and Pack, J D. "Application of Least Squares Lattice Algorithms to Adaptive Equalization", IEEE Trans., 1981, COM-29, p 136-142.
- 15 Satorius, E H., and Pack, J D. "A Least Squares Adaptive Lattice Equalizer Algorithm", Naval Ocean Systems Center, Tech. Report 575, Sept 1980.

- 16 Satorius, E H., and Shensa, M J. "On the Application of Recursive Least Squares Methods to Adaptive Processing", Proc. Int. Workshop on Applications of Adaptive Control, Yale Univ., New Haven, CT., August 1979, p 165-191.
- 17 Ahmed, N., Hummels, D R., Uhl, M L., and Soldan, D L., "A Short-Term Sequential Regression Algorithm", IEEE Trans., 1979, ASSP-27, p 453-457.



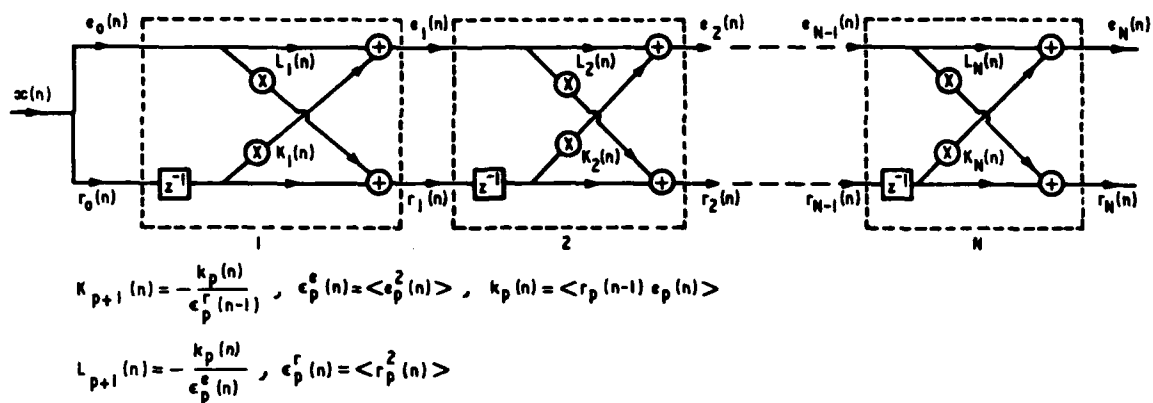


Figure 1 N-stage linear prediction lattice configuration suitable for non-stationary statistics in data sequence  $\{x(n)\}$ .

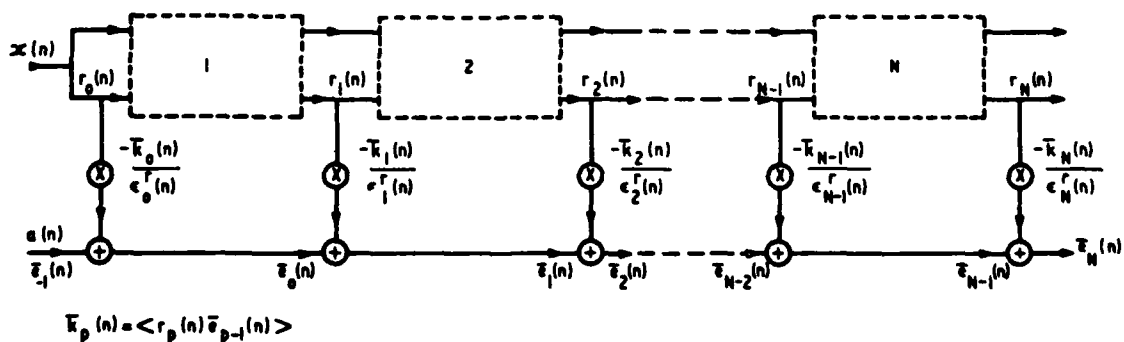


Figure 2 Extension of lattice filter to L-stage equaliser configuration (L = N+1).



## DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... **UNCLASSIFIED** .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S) )

1. DRIC Reference (if known)	2. Originator's Reference <b>MEMORANDUM 3567</b>	3. Agency Reference	4. Report Security Classification <b>UNCLASSIFIED</b>	
5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location <b>ROYAL SIGNALS AND RADAR ESTABLISHMENT</b>			
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location			
7. Title <b>A COMPARISON BETWEEN THE LEAST-SQUARES LATTICE AND FAST KALMAN ALGORITHMS FOR ADAPTIVE CHANNEL EQUALISATION</b>				
7a. Title in Foreign Language (in the case of translations)				
7b. Presented at (for conference papers)    Title, place and date of conference				
8. Author 1 Surname, initials <b>McWHIRTER, J.G.</b>	9(a) Author 2 <b>SHEPHERD, T.J.</b>	9(b) Authors 3,4...	10. Date	pp. ref.
11. Contract Number	12. Period	13. Project	14. Other Reference	
15. Distribution statement <b>UNLIMITED</b>				
Descriptors (or keywords)				
continue on separate piece of paper				
Abstract <b>An exact least-squares lattice algorithm for channel equalisation is derived without using an explicit time update for the equaliser or predictor coefficients. This avoids the need for auxiliary vectors and scalars within the analysis and facilitates a theoretical comparison with the corresponding Fast Kalman algorithm which is also derived. Both algorithms incorporate a very general form of statistical estimator which includes the growing or growing-fading memory estimators and the sliding window estimator as special cases.</b>				

DATE  
ILME